Università degli Studi dell'Insubria
Facoltà di Scienza MM.FF.NN
Corso di laurea in Informatica

# Realizzazione di uno strumenti per la gestione ed organizzazione dei tempi di esecuzione di JTabwb

Tesi di Laurea di Eleonora Altana
Num. di matricola 609197
Relatore: Mauro Ferrari

# Indice

# Introduzione

Il progetto di questa tesi ha lo scopo di gestire ed elaborare i tempi di esecuzione del sistema JtabWb.
Si  tratta quindi di una fase di caricamento dei tempi di esecuzione e successivamente effettuare interrogazioni per analizzare i dati.
L'applicazione per effettuare queste operazioni e' stata realizzata utilizzando JavaFx 8.

# JavaFX

JavaFX è un insieme di pacchetti graficia e multimediali che consente agli sviluppatori di progettare,
creare, testare, eseguire il debug e distribuire applicazioni rich client che operano in modo coerente nelle diverse piattaforme.
Dal momento che la libreria JavaFX è scritto come un API Java, il codice dell'applicazione JavaFX può fare riferimento ad API di qualsiasi libreria Java.
Ad esempio, le applicazioni JavaFX possono utilizzare librerie Java API per accedere a funzionalità native del sistema.
L'aspetto grafico delle applicazioni JavaFx e' personalizzabile: si possono ad esempio usare fogli di stile (css) oppure utilizzare il linguaggio di scription fxml per separare la parte di layout grafico da quella di sviluppo delle funzionalita'.
Le Api JavaFx sono integrate nel Java SE Runtime Environment (JRE) e nel Java Development Kit (JDK).
Dato che JDK e' disponibile per la maggior parte delle piattaforme desktop (Windows, Mac OS e Linux), le applicazioni JavaFx compilate con JDK 8 possono essere eseguite sulla maggior parte delle piattaforme.

## Architettura

Questo paragrafo fornisce una descrizione ad alto livello dell'architettura di JavaFx.

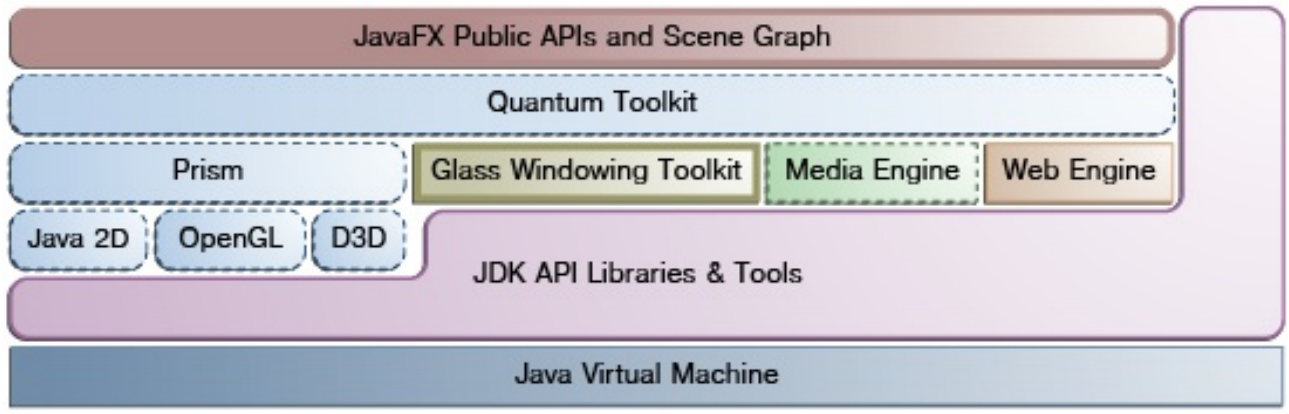


*Figura 1.1 - Diagramma architettura JavaFX*

### Scene Graph

Scene graph, nella parte alta dell'architettura, e' il punto dipartenza per costruire applicazioni JavaFx. E' una gerarchia ad albero di nodi che rappresentano tutti gli elementi visivi dell'interfaccia utente.

Un singolo elemento nel scene graph e' chiamato nodo.

Ogni nodo possiede un ID, una class per lo stile e dimensioni limitate.

Fatta eccezione per il nodo principale dello scene graph (root node), ogni nodo ha un unico genitore e zero o piu' figli.

Ogni nodo possiede inoltre le seguenti caratteristiche:

- Effetti (blur, shadows)
- Opacita'
- Trasformazioni
- Gestori di eventi

Il JavaFx scene graph include inoltre primitive grafiche, come rettangoli e testo, in modo da avere container di layout, immagini e media

Le API `javafx.scene` permettono di creare e specificare  diversi tipi di contenuti, come:

- Nodi : forme (2-D e 3-D), immagini, testi, browser web, grafici, gruppi, contenitori
- Stati: trasformazioni (posizionamento e orientamento dei nodi), effetti visivi ed altri stati visivi dei contenuti
- Effetti: semplici oggetti che cambiano l'aspetto del nodo, come l'effetto sfocato (blur), l'ombreggiatura (shadow) o le modifiche di colore

## Java public APIs for JavaFX

Il livello piu' alto dell'architettura JavaFX contiene una set completo di api Java che supportano lo sviluppo di applicazioni rich client.

Queste APIs forniscono liberta' e flessibilita' nel costruire applicazioni rich client, dando a JavaFX le seguenti caratteristiche:

- permettono di poter utilizzare le piu' potenti caratteristiche Java, come i tipi generici, annotazioni, multithreading ed espressioni Lamda
- rendono semplice per uno sviluppatore web che utilizzano altri linguaggi basati su JVM come Groovy o Javascript usare JavaFX
- permettono ad uno sviluppatore Java di usare altri linguaggi di sistema, come Groovy, per scrivere ampie e complesse applicazioni JavaFX
- estende la libreria Java includendo le observable list e maps, che permettono alle applicazioni di collegare interfaccia utente a modelli dati, osservarne i cambiamenti e aggiornare l'interfaccia

Molte delle APIs JavaFX sono state incluse direttamente in Java. Alcumen APIs, come Layout e Media, sono state migliorate e semplificate basandosi sul feedback ricevuto dagli utenti della release di JavaFx 1.x.

## Graphics System

La JavaFX Graphics System, si trova sotto il livello scene graph.

Supporta sia 2-D che 3-D ed e' provvisto di software per il rendering che iterviene quando l'hardware grafico del sistema e' insufficiente.

Ci sono due pipeline grafiche implementate nella piattaforma JavaFX:

- **Prism**, che puo' essere eseguito sia su software che hardware per il rendering, incluso il 3-D. Si occupa di effettuare il rendering delle JavaFx scenes.

  I render sono possibili in base al dispositivo in uso:

  - DirectX 9 su Windows XP e Windows Vista
  - DirextX 11 su Windows 7
  - OpenGL su Mac, Linux, Embedded
  - rendering software quando l'accelerazione hardware non e' possibile

    l'accelerazione hardware e' usata quando e' possibile, ma quando non e' disponibile, e' usato il render software perche' e' distribuito in tutte le Java Runtime Environments

## Glass Windowing Toolkit

Il glass windonwing toolkit, visualizzato in grigio al centro della figura 1.1, e' il livello piu' basso nello stack grafico di JavaFx. La sua principale funzionalita' e' quella di fornire un servizio operativo nativo, come gestire le finestre, i timer, etc.

Serve per connettere la piattaforma JavaFx al sistema operativo nativo.

Il glass windowing tollkit e' anche responsabile della gestione della coda di eventi: utilizza la coda di eventi nativa del sistema operativo per definire l'uso dei thread.

Il glass toolkit e' eseguito sullo stesso thread dell'applicazione JavaFx

### *Threads*

Il sistema esegue uno o piu' dei seguenti thread in un dato momento

- **applicazione JavaFx** : e' il thread primario utilizzato dagli sviluppatori di applicazioni JavaFx. Ogni scena “live”, che rappresenta una parte di una finestra, deve essere raggiungibile da questo thread. La scena grafica puo' essere creata e manipolata in un thread eseguito in background, ma quando il suo nodo root e' collegato ad un qualunque oggetto oggetto “attivo” nella scena, questa scena grafica deve essere raggiungibile dal thread dell'applicazione JavaFX.  Questo permette agli sviluppatori di realizzare scene grafiche complesse in un thread in background, mantenendo le animazioni della scena “live” fluide.

- **Render Prism**: questo thread gestisce il rendering in modo indipendente dall gestore degli eventi. Permette di visualizzare ilframe N fino a che il frame N+1 viene processato. Questa abilita' di avere processi concorrenti e' un grande vantaggio, specialmente sui moderni sistemi operativi che dispongono di multi-processore.
- **Media**: questo thread viene eseguito in background e sincronizza l'ultimo frame attraverso la scena grafica usanto il thread dell'applicazione JavaFX

### *Pulse*

Un impulso (pulse) e' un evento che indica alla scena grafica di JavaFX che e' il momento di sincronizzare lo stato degli elementi sulla scena con Prism. L'impulso e' limitato a 60 frames al secondo (fps) e viene emesso ogni volta che sulla scena sono in esecuzione animazioni. Anche se non ci sono animazioni, e' schedulato un impulso quando qualcosa nella scena e' cambiato, come ad esempio la posizione di un bottone.

Quando viene emesso l'impulso, lo stato degli elementi della scena grafica e' sincronizzato fino al livello di render. Un impulso mette a disposizione degli sviluppatori un modo per gestire eventi in modo asincrono, permettendo al sistema di preparare ed eseguire eventi al momento dell'impulso.

Anche il layout e il CSS sono legati all'impulso. Numerosi cambiamenti nella scena grafica possono possono portare ad avere layout multipli  o aggiornamenti CSS, con la possibilita' di degradare le performance.

Per mantenere le prestazioni, il sistema permette di lasciare passare la sincronizzazione di una sola modifica ad impulso.

Media